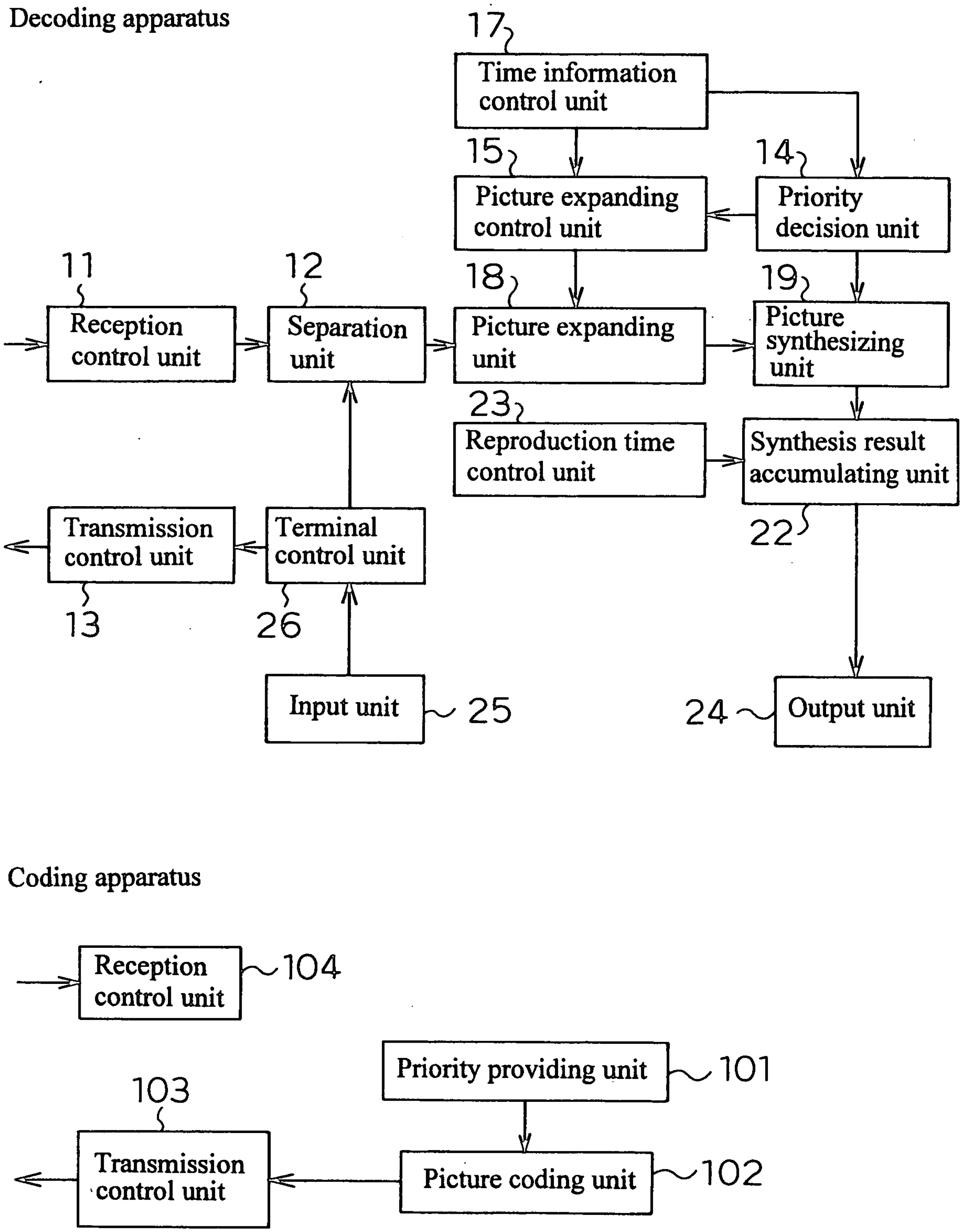
Fig. 1

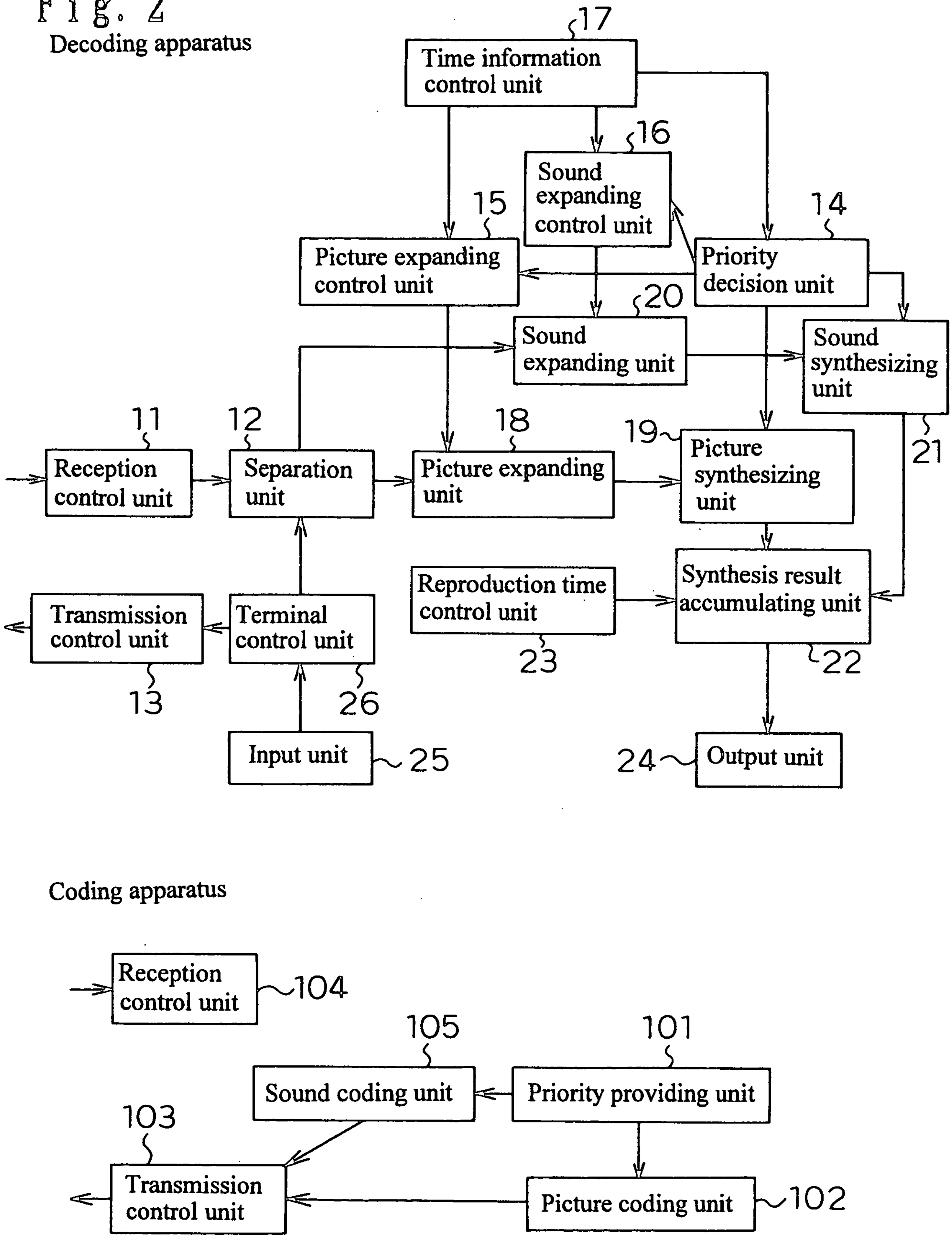


Decoding apparatus

17
Time information control unit
15
Picture expanding control unit
14
Priority decision unit
11
Reception control unit
12
Separation unit
18
Picture expanding unit
19
Picture synthesizing unit
23
Reproduction time control unit
Synthesis result accumulating unit
22
Transmission control unit
Terminal control unit
13
26
Input unit
25
24
Output unit

Coding apparatus

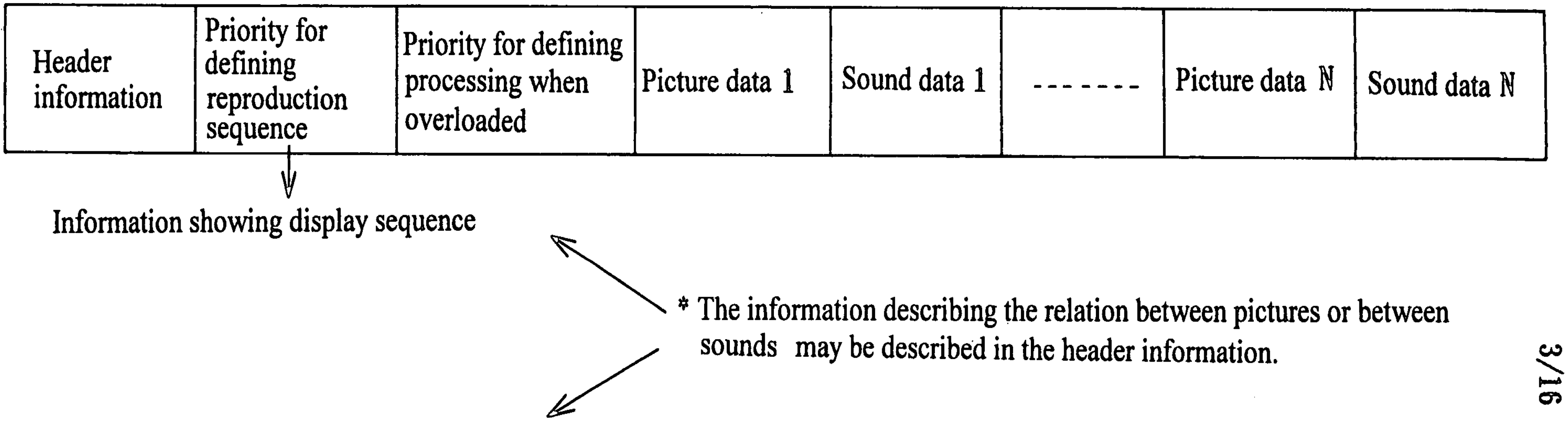
Fig. 2


Decoding apparatus

Coding apparatus

Fig. 3 (a)

All multiplexed format



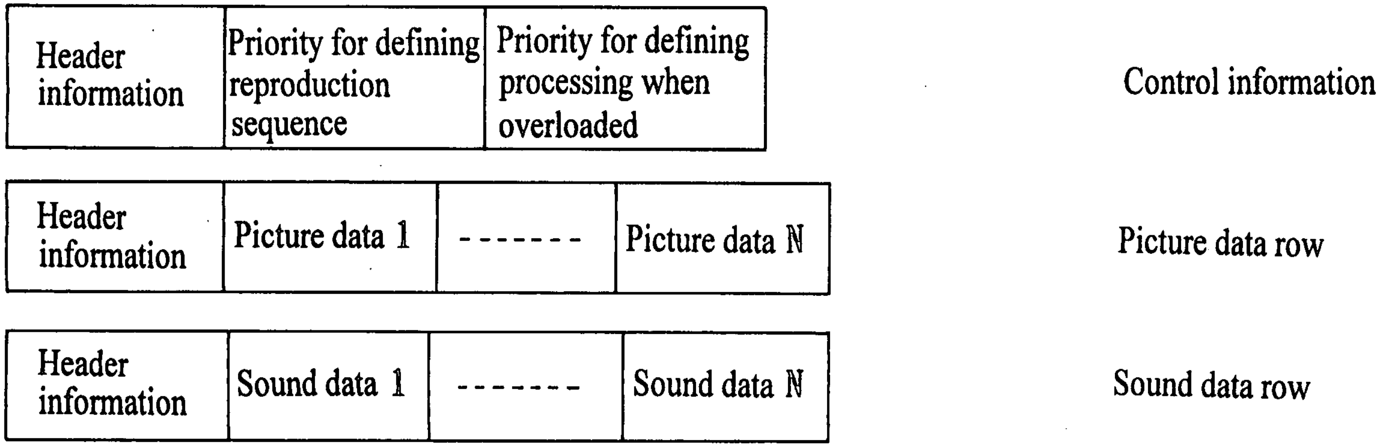
Header information	Priority for defining reproduction sequence	Priority for defining processing when overloaded	Picture data 1	Sound data 1	-------	Picture data N	Sound data N

Information showing display sequence

* The information describing the relation between pictures or between sounds may be described in the header information.

Fig. 3 (b)

Multiplexed in individual media, and transmitted from respective communication ports



Header information	Priority for defining reproduction sequence	Priority for defining processing when overloaded		Control information
Header information	Picture data 1	-------	Picture data N	Picture data row
Header information	Sound data 1	-------	Sound data N	Sound data row

Fig. 4

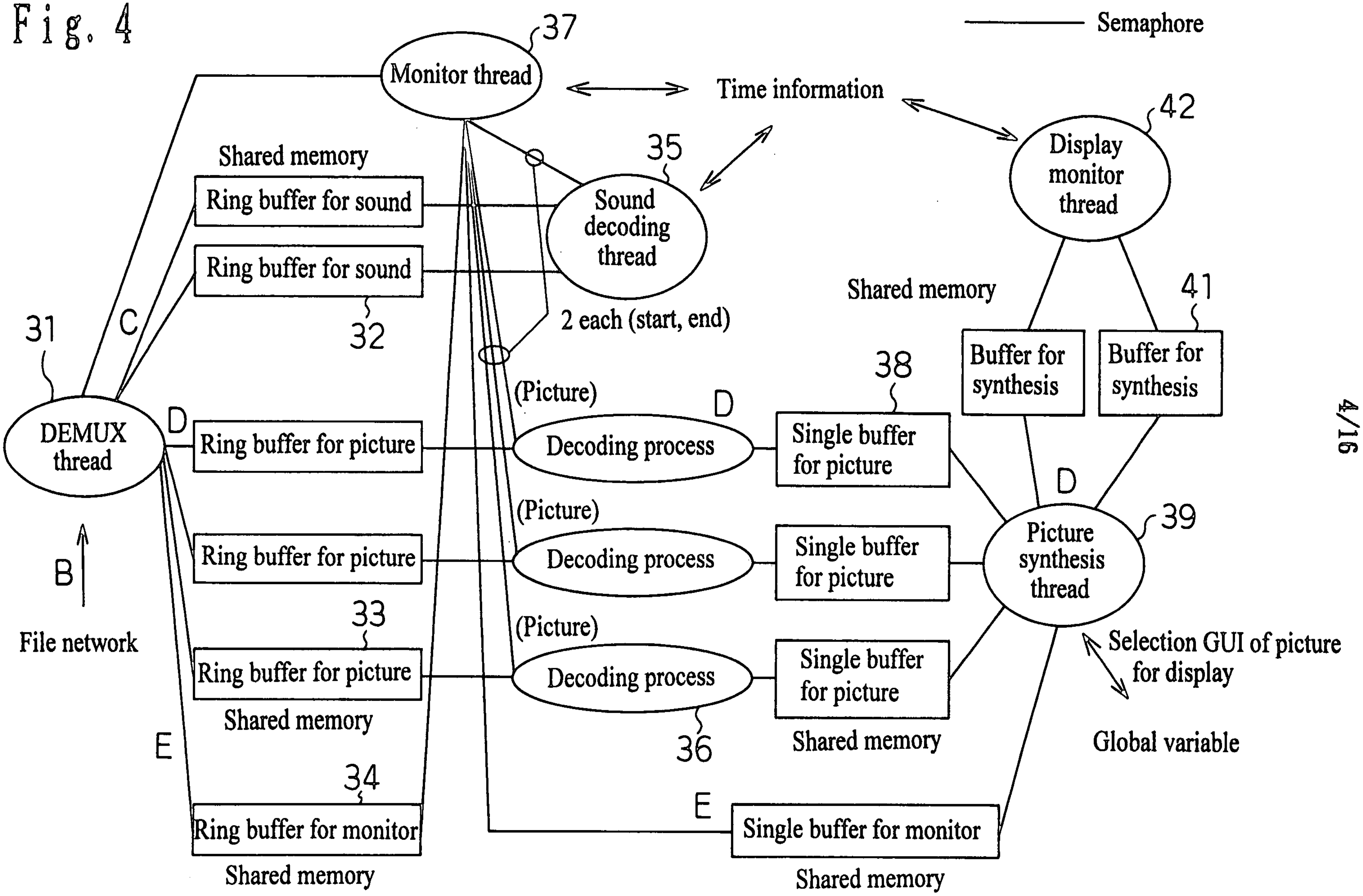

Semaphore
37
Monitor thread
Time information
42
Display monitor thread
Shared memory
Ring buffer for sound
Ring buffer for sound
32
35
Sound decoding thread
2 each (start, end)
Shared memory
41
Buffer for synthesis
Buffer for synthesis
31
C
DEMUX thread
D
Ring buffer for picture
(Picture)
Decoding process
D
38
Single buffer for picture
D
39
Picture synthesis thread
B
File network
Ring buffer for picture
(Picture)
Decoding process
Single buffer for picture
33
Ring buffer for picture
(Picture)
Decoding process
Single buffer for picture
Selection GUI of picture for display
Shared memory
36
Shared memory
Global variable
E
34
Ring buffer for monitor
E
Single buffer for monitor
Shared memory
Shared memory

Fig. 5

B.

```
struct shm_tspkt  {
      data_byte        188byte          Packet data
}
```

C.

```
struct shm_apkt  {
    DWORD  sync_code          32bit  Packet synchronous code
    WORD   pts                16bit  Display time
    WORD   frame_length       16bit  Frame length
    BYTE   data_byte          Nbyte  Sound data
                                      (N=frame_length)
}
```

D.

```
struct shm_vpkt  {
    DWORD  sync_code           32bit  Packet synchronous code
    BYTE   temporal_reference  8bit   Frame number
    WORD   frame_length        16bit  Frame length
    BYTE   data_byte           Nbyte  Picture data
                                       (N=frame_length)
}
```

E.

```
struct shm_kanshi_lnfo  {
    WORD  pts                      16bit    Display time
    BYTE  number_of_object         8bit     No. of objects
    for (i=0:  i<number_of_object:i++)        {
      BYTE   object_id              8bit         ID
      BYTE   temporal_reference     8bit         Frame number
      BYTE   object_priority        4bit         Priority (*1)
             reserved               2bit
             IPB_flag               2bit         Frame type
      WORD   horizontal_offset      10bit   Display position, horizontal
      WORD   vertical_offset        10bit   Display position, vertical
      BYTE   layer                  4bit         Layer
    }
}
```

(*1) Bits are assigned from the highest position sequentially by 4 bits (object_priority), 2 bits, 2 bits (IPB_flag)

Fig. 6

DEMUX thread

```
void demux ( )
{
      Shared memory (ring), semaphore generation process: for output
                (2 for sound, 3 for picture, 1 for monitor table)
      Semaphore generation for monitor thread control (one)
      BOOL   flag = TRUE:  //   State of ring buffer

      while(1) {
            if(flag) Reading from file or network                          -(5-1)

            if(flag)
                 Analysis of 188-byte packet data, setting in specified structure   -(5-2)
                   (decomposed into information of sound, picture, monitor table)

            //  Exclusive control of ring buffer by semaphore
            if  (Able to write in ring buffer ?){
                 Write into ring buffer (from earlier object ID, write sequentially   -(5-3)
                               into shared  memory of earlier buffer number)
                 Advance write pointer of written buffer                     -(5-4)
                 flag=TRUE:
            }else
                 flag=FALSE:  //   Prevent overflow of ring buffer

            if(flag)
                 After writing information of pictures and sounds for one monitor  -(5-5)
                 table, advance  the counter of semaphore for monitor thread control
      }
}
```

Fig. 7

Monitor thread

```
void Watch Process ( )
{
      BYTE disp_TR[i]:  //  Picture serial number (shared memory)
      BOOL skip_flag[i]:  //  Skip flag to which decoding process refers
                                  (shared memory)

     Shared memory (ring buffer: monitor table 1)
                        Semaphore open: used by determining priority of processing
     Shared memory (single buffer: monitor table 1)
                        Semaphore generation: transfer to synthesis side
     Generation of semaphore for process monitor
     Semaphore open for monitor thread control (one)

     Start of picture decoding process
     Confirm start of process
     while {skip_flag[i]=FALSE:  //  Not skipped  }

     while(1)
     {
          Reading of monitor table (read pointer update, from DEMUX)
          Check of object priority                          -(6-1)  -(6-2)
          Writing of monitor table (to synthesis side)               -(6-3)
          Wait for creation of data for one monitor table from DEMUX -(6-4)

          From highest priority
          {
             disp_TR[i]=TR:                                          -(6-5)
             if  ( Present time > display time (pts) )  {            -(6-6)
                       Not skipped if I frame
                       skip_flag[i]=FALSE
             }else{
                       P, B frames are skipped
                       skip_flag[i]=TRUE
             }
            Release of semaphore of corresponding process            -(6-7)
            Wait for release of semaphore of corresponding process   -(6-8)
                                        (process completion check)
          }
     }
}
```

Fig. 8

Decoding process

```
void main(int argc, char *argv[ ] )
{

    Value received from main process :
         Shared memory to be opened, name of semaphore

    Shared memory (ring), open processing of semaphore: for input (from MUX)
    Shared memory (single), open processing of semaphore: for output (to synthesis
                                                                      side)

    while(1)  {
         Monitor thread waits for release of semaphore                  - (7-1)

         Input picture state check:                                     - (7-2)
                 Picture serial number (TR), skip input frame?

         Wait for picture data to be decoded                            - (7-3)

         Is TR present in shared memory?  {                             - (7-4)
              Skip decoding if not present
              Advance read pointer for ring buffer (for input)
         }

         if  (Skip one input frame)  {
                 Decoding process                                       - (7-5)
                 Advance read pointer for ring buffer (for input)
         }

         Output of decoding result (*1)                                 - (7-6)
         Release semaphore to monitor thread (process end notice)       - (7-7)
    }
}
```

(*1) When skipping input frame process, send signal to main process without decoding process and output of decoding result

Fig. 9

Picture synthesis thread

```
void Watch Sync ( )
{
    Shared memory (single), semaphore generation process: for input (from decoder)
    Shared memory (single), semaphore generation process: for input (from monitor thread)
    Shared memory (single), semaphore generation process: for output (to display monitor: 2)
    BOOL flag=TRUE:

    while(1)   {
        Wait for monitor table from monitor thread                  - (8-1)
        Check priority order of object                              - (8-2)

        From highest priority order  {                              - (8-3)
            Wait for picture of decoding result (accumulated in shared memory)
            //  Totally black if empty
        }
        Synthesis of image adjusting to display position            - (8-4)

        //   Double buffer
        if (flag)   {                                               - (8-5)
            Write synthesis result into shared memory (to display monitor) #1
            flag=FALSE:
        } else  {
            Write synthesis result into shared memory (to display monitor)#2
            flag=TRUE:
        }
    }
}
```

Fig. 10

Display monitor thread

```
void Watch Disp ( )
{
    Shared memory (single), open processing of semaphore: for input
                                                    (from synthesis  thread: 2)
    BOOL   flag = TRUE:


     while(1)
    {
          // Double buffer
          if (flag)   {
               Wait for synthesis picture from shared memory (from synthesis thread)#1
               flag = FALSE:                                                  -(9-1)
          }  else  {
              Wait for synthesis picture from shared memory (from synthesis thread)#2
               flag = TRUE:
          }

          if  (Initial display)    {
               Acquire display start time from timer                          -(9-2)
          }
          Sleep  (pts-nowtime):                                               -(9-3)
            Display of synthesis picture
    }
}
```

Fig. 11

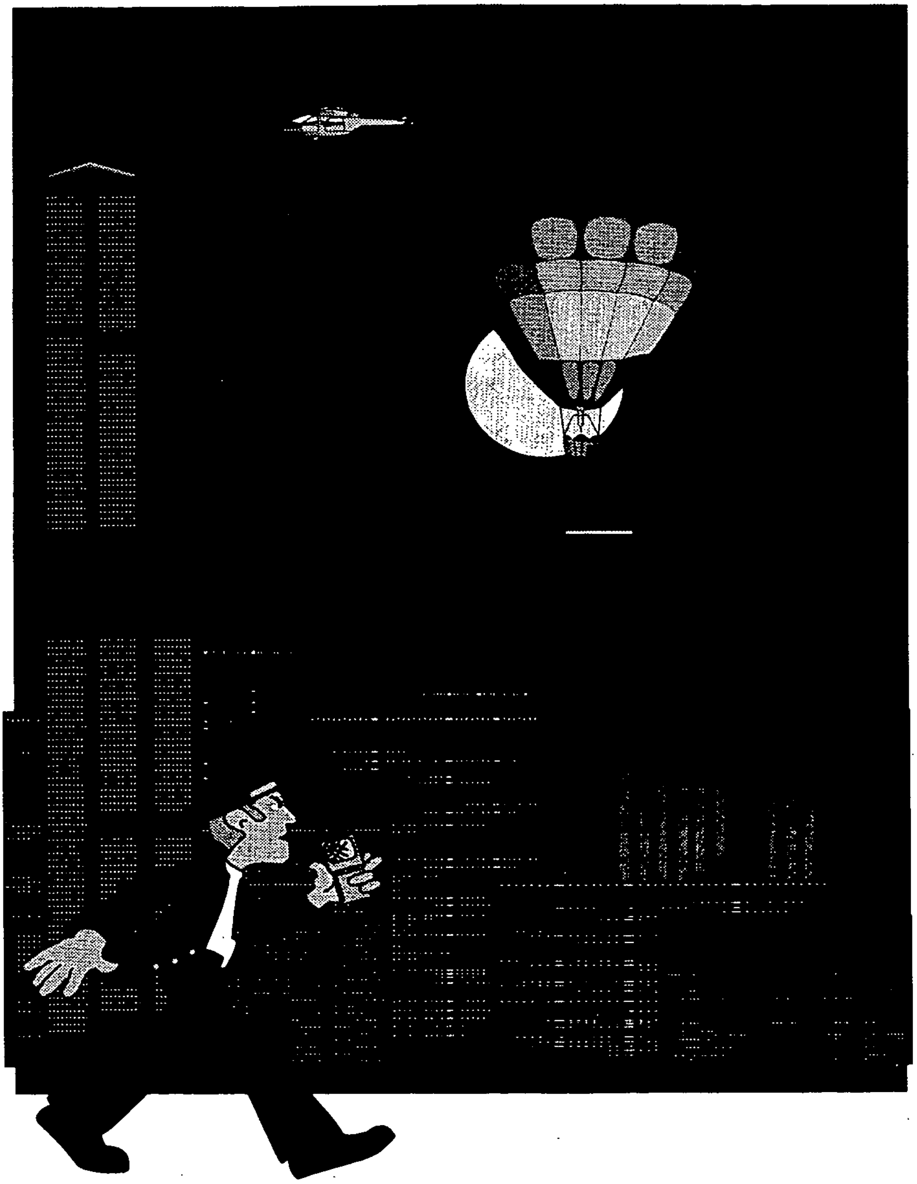

Three-dimensional picture
(foreground: helicopter)
Three-dimensional picture
(foreground: balloon)
Background picture
(night sky)
Foreground picture
(building)
Synthesis ratio: 0.5
Foreground picture (man)

Fig. 12 (a)

System of hardware base

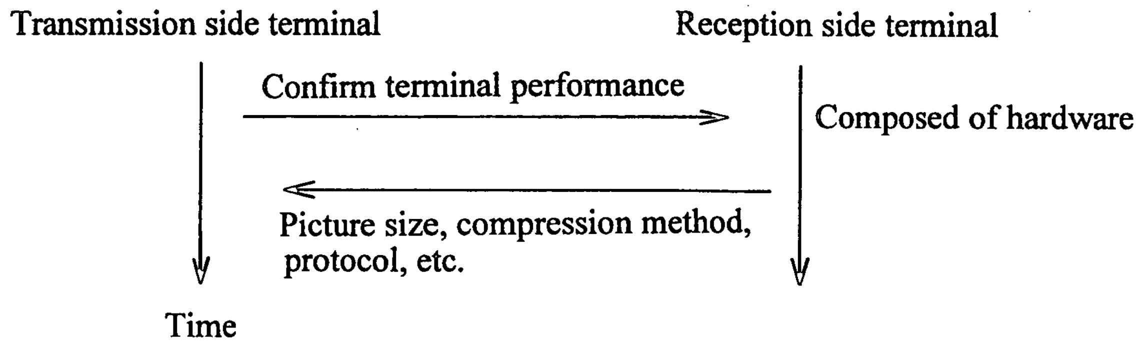

Transmission side terminal
Reception side terminal
Confirm terminal performance
Composed of hardware
Picture size, compression method, protocol, etc.
Time

Fig. 12 (b)

System of software base

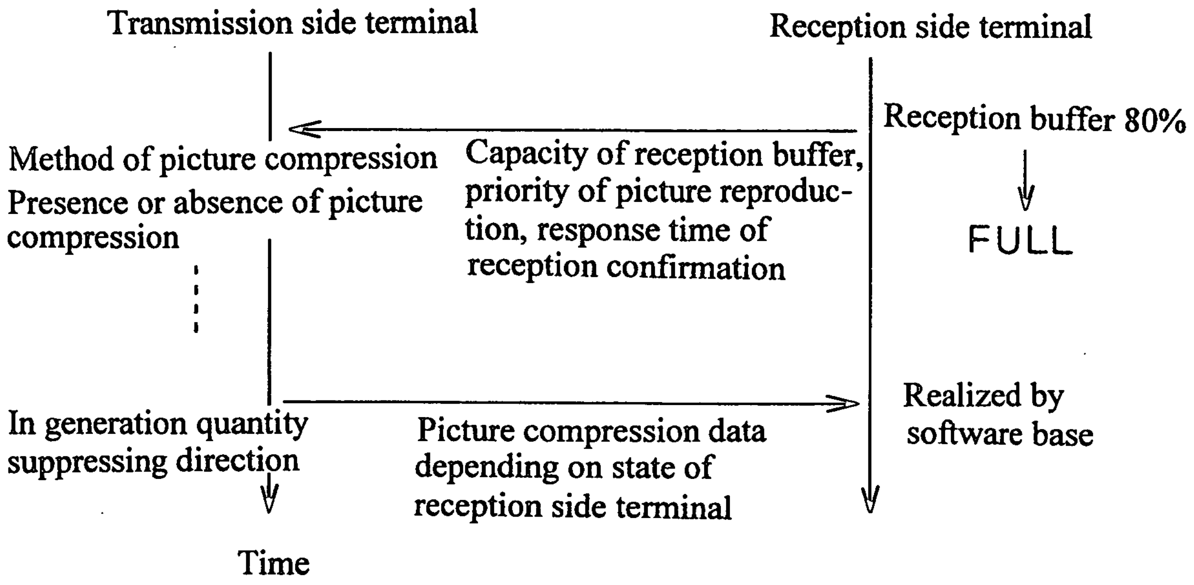
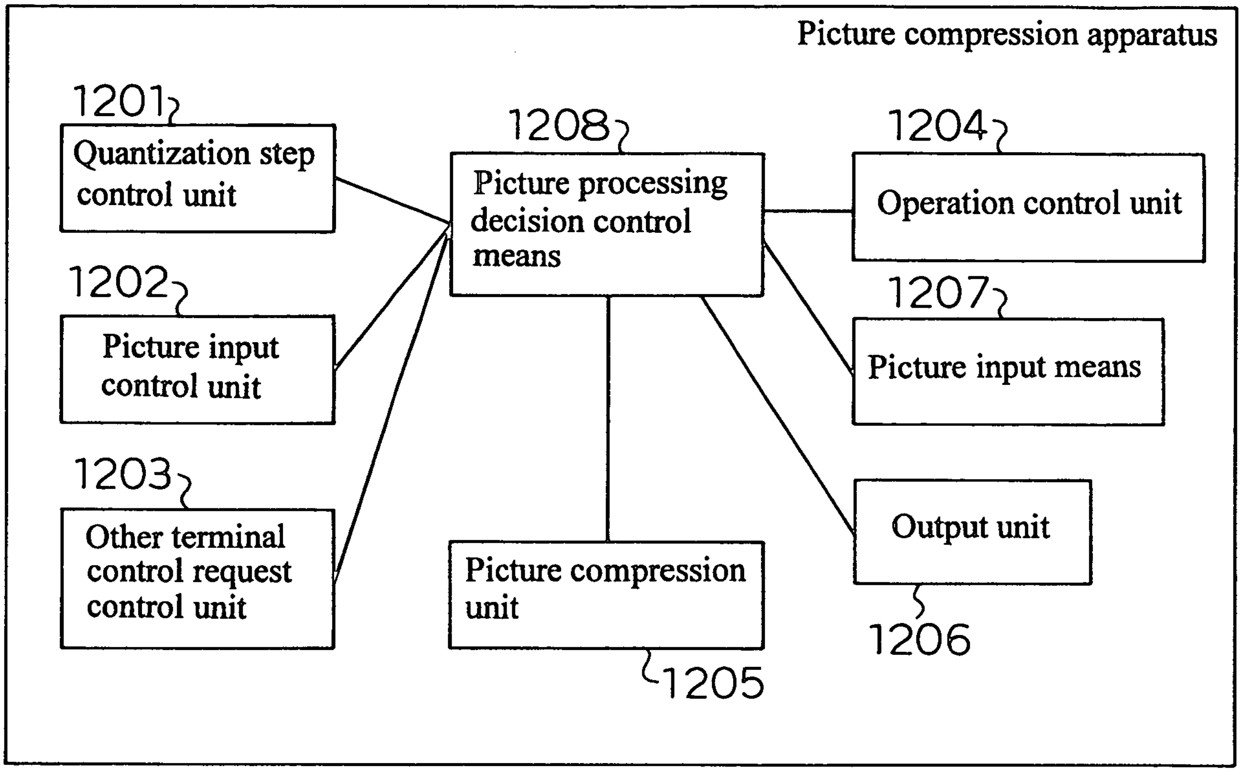

Transmission side terminal
Reception side terminal
Reception buffer 80%
FULL
Method of picture compression
Presence or absence of picture compression
Capacity of reception buffer, priority of picture reproduction, response time of reception confirmation
Realized by software base
In generation quantity suppressing direction
Picture compression data depending on state of reception side terminal
Time

Fig. 13


Picture compression apparatus
1201
Quantization step control unit
1202
Picture input control unit
1203
Other terminal control request control unit
1208
Picture processing decision control means
1204
Operation control unit
1207
Picture input means
Output unit
1206
Picture compression unit
1205

* Sound compression apparatus can be set similarly

Fig. 14

Picture size	Camera control	Other terminal control request	Quantization step
QCIF	Pan	Buffer over	16
CIF	None	None	16
QCIF	None	None	18
QCIF	Tilt	None	14
⋮	⋮	⋮	⋮

Fig. 15

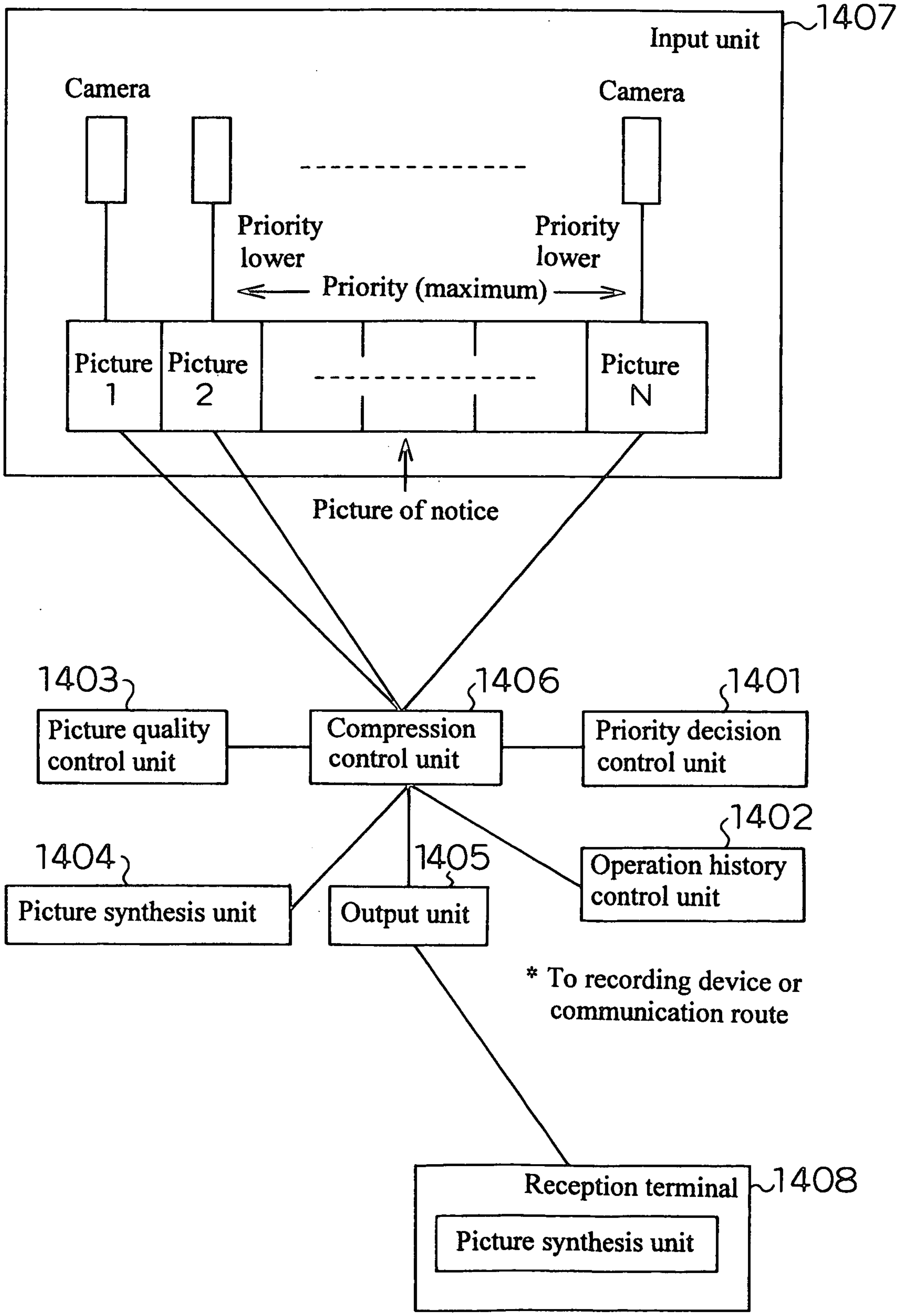
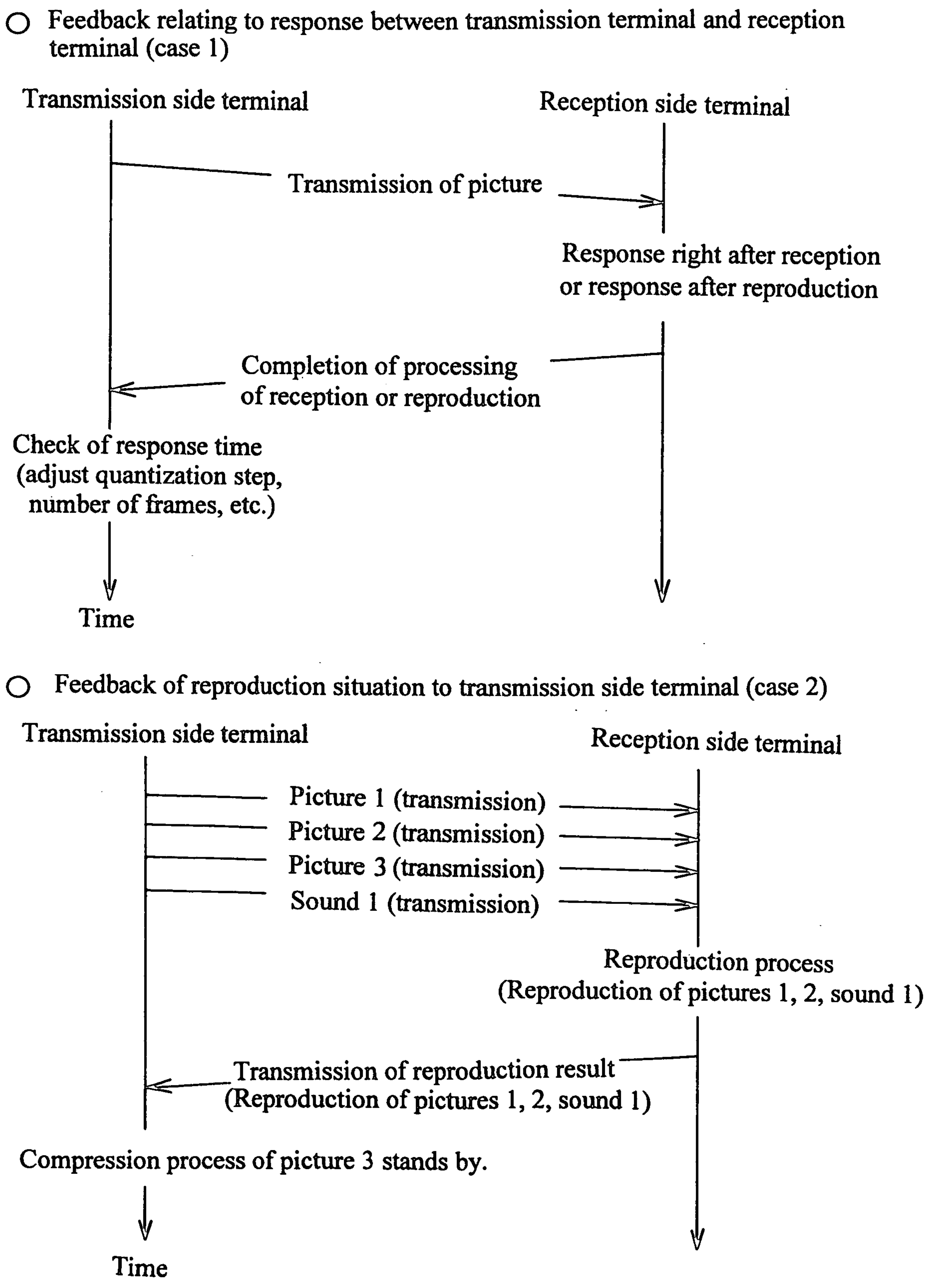

1407
Input unit
Camera
Camera
Priority
lower
Priority
lower
Priority (maximum)
Picture
1
Picture
2
Picture
N
Picture of notice
1403
Picture quality
control unit
1406
Compression
control unit
1401
Priority decision
control unit
1402
Operation history
control unit
1404
Picture synthesis unit
1405
Output unit
* To recording device or
communication route
Reception terminal
1408
Picture synthesis unit

Fig. 16



○ Feedback relating to response between transmission terminal and reception terminal (case 1)

Transmission side terminal
Reception side terminal
Transmission of picture
Response right after reception or response after reproduction
Completion of processing of reception or reproduction
Check of response time (adjust quantization step, number of frames, etc.)
Time

○ Feedback of reproduction situation to transmission side terminal (case 2)

Transmission side terminal
Reception side terminal
Picture 1 (transmission)
Picture 2 (transmission)
Picture 3 (transmission)
Sound 1 (transmission)
Reproduction process (Reproduction of pictures 1, 2, sound 1)
Transmission of reproduction result (Reproduction of pictures 1, 2, sound 1)
Compression process of picture 3 stands by.
Time